Installing & Running the HPCC Platform

Boca Raton Documentation Team
Welcome

These instructions will guide you through installing and running the HPCC\(^1\) Community Edition on a single node to start and then optionally, expand it to a larger cluster of nodes.

The HPCC Thor technology is designed to effectively process, analyze, and find links and associations within high volumes of complex data. This can detect non-obvious relationships, scale to support petabytes of data, and is significantly faster than competing technologies while requiring less hardware and resources.

The HPCC Roxie technology - also known as the Rapid Data Delivery Engine or RDDE - uses a combination of technologies and techniques that produce extremely fast throughput for queries on indexed data.

This translates into better quality answers in less time so that organizations can cope with massive data and efficiently turn information into knowledge.

We suggest reading this document in its entirety before beginning. The entire process can take an hour or two, depending on your download speed.

\(^1\)High Performance Computing Cluster (HPCC) is a massively parallel processing computing platform that solves Big Data problems. See http://hpccsystems.com/Why-HPCC/How-it-works for more details.
Hardware and Software Requirements

The following section describes the various hardware and software required in order to run the HPCC.

**Network Switch**

A significant component of HPCC is the infrastructure it runs on, specifically the switch.

**Switch requirements**

- Sufficient number of ports to allow all nodes to be connected directly to it;
- IGMP v.2 support
- IGMP snooping support

**Small:** For a very small test system, almost any gigabit switch will suffice. These are inexpensive and readily available in six to 20-port models.

**Figure 1. 1 GigE 8-port Switch**

![1 GigE 8-port Switch](image1)

**Medium:** For medium sized (10-48 node) systems, we recommend using a Force10 s25, s50, s55, or s60 switch

**Figure 2. Force10 S55 48-port Network Switch**

![Force10 S55 48-port Network Switch](image2)
Large: For large (48-350 node) system, the Force10 c150 or c300 are good choices.

Figure 3. Force 10 c150
Very Large: For very large (more than 300 nodes) system, the Force10 e600 or e1200 are good choices.

Figure 4. Force 10 e600 and e1200

Switch additional recommended features

- Non-blocking backplane
- Low latency (under 35usec)
- Layer 3 switching
- Managed and monitored (SNMP is a plus)
- Port channel (port bundling) support
Load Balancer

In order to take full advantage of a Roxie cluster, a load balancer is required. Each Roxie Node is capable of receiving requests and returning results. Therefore, a load balancer distributes the load in an efficient manner to get the best performance and avoid a potential bottleneck.

We recommend the Web Accelerator product line from F5 Networks. See http://www.f5.com/pdf/products/big-ip-webaccelerator-ds.pdf for more information.

Figure 5. F5 Load Balancers

Load Balancer Requirements

Minimum requirements

• Throughput: 1Gbps Gigabit
• Ethernet ports: 2
• Balancing Strategy: Round Robin

Standard requirements

• Throughput: 8Gbps
• Gigabit Ethernet ports: 4
• Balancing Strategy: Flexible (F5 iRules or equivalent)

Recommended capabilities

• Ability to provide cyclic load rotation (not load balancing).
• Ability to forward SOAP/HTTP traffic
• Ability to provide triangulation/n-path routing (traffic incoming through the load balancer to the node, replies sent out the via the switch).
• Ability to treat a cluster of nodes as a single entity (for load balancing clusters not nodes)

or

• Ability to stack or tier the load balancers for multiple levels if not.
Nodes-Hardware

The HPCC can run as a single node system or a multi node system.

These hardware recommendations are intended for a multi-node production system. A test system can use less stringent specifications. Also, while it is easier to manage a system where all nodes are identical, this is not required. However, it is important to note that your system will only run as fast as its slowest node.

**Node mandatory requirements**

- Pentium 4 or newer CPU
- 64-bit
- 4GB RAM
- Two Hard Drives (with sufficient free space to handle the size of the data you plan to process)
- 1 GigE network interface

**Node recommended specifications**

- Nehalem Core i7 CPU
- 64-bit
- 4 GB RAM (or more)
- 1 GigE network interface
- PXE boot support in BIOS

PXE boot support is recommended so you can manage OS, packages, and other settings when you have a large system

- Optionally IPMI and KVM over IP support

**For Roxie nodes:**

- Two 10K RPM (or faster) SAS Hard Drives

  Typically, drive speed is the priority for Roxie nodes

**For Thor nodes:**

- Two 7200K RPM (or faster) SATA Hard Drives
- Optionally 3 or more hard drives can be configured in a RAID 5 container for increased performance and availability

  Typically, drive capacity is the priority for Thor nodes
Nodes-Software

All nodes must have the identical operating systems. We recommend all nodes have identical BIOS settings, and packages installed. This significantly reduces variables when troubleshooting. It is easier to manage a system where all nodes are identical, but this is not required.

Operating System Requirements

- 64-bit LINUX CentOS 5.x / Red Hat
- 64-bit Ubuntu 10.04 LTS or 64-bit Ubuntu 11.04
- 64-bit OpenSuSe 11.3 and 11.4

Dependencies and packages

Running HPCC depends on having these required component packages listed below installed on the system. For details of the installation commands recommended for obtaining these packages, see Specific OS node software installation commands in the Appendix.

CentOS:

- Libicu
- The GNU Compiler Collection (GCC) with G++ support
- Xalan-c 1.10.0 (xalan-c-1.10.0-2.el5.x86_64.rpm -- available from the EPEL project)
- Xerces-c 2.7.0 (xerces-c-2.7.0-8.el5.x86_64.rpm)
- Boost 1.41 (boost141-regex-1.41.0-2.el5.x86_64.rpm available from the EPEL project)
- openssh-server & openssh-clients
- OpenSSL libraries (if optional HPCC SSL support is used)
- Openldap
- m4
- expect
- libtool

Ubuntu:

- libcud-dev
- libboost-regex-dev (1.41 available from the EPEL project)
- The GNU Compiler Collection (GCC) with G++ support
- Xalan-c 1.10.0

  libxalan110

- Xerces-c 2.8.0

  libxerces-c28

- binutils
- libldap2-dev
- openssl
- zlib1g
- expect
- openssh-client
- openssh-server
SuSe:

- Libicu
- The GNU Compiler Collection (GCC) with G++ support
- Xalan-c 1.10.0
- Xerces-c 3.0
- openssh
- libboost 1.41
- libldap
- expect
SSH Keys

The HPCC components use ssh keys to authenticate each other. This is required for communication between nodes. A script to generate keys has been provided. You should run that script and distribute the public and private keys to all nodes after you have installed the packages on all nodes, but before you configure a multi-node HPCC.

• As root or using sudo, generate a new key using this command:

```
/opt/HPCCSystems/sbin/keygen.sh
```

• Distribute the keys to all nodes. From the `/home/hpcc/.ssh` directory, copy these three files to the same directory (`/home/hpcc/.ssh`) on each node:
  • `id_rsa`
  • `id_rsa.pub`
  • `authorized_keys`

Make sure that files retain permissions when they are distributed. These keys need to be owned by the user "hpcc".
User Workstation Requirements

- Running the HPCC platform requires communication from your user workstation with a browser to the HPCC. You will use it to access ECL Watch—a Web-based interface to your HPCC system. ECL Watch enables you to examine and manage many aspects of the HPCC and allows you to see information about jobs you run, data files, and system metrics.

  Use one of the supported web browsers with Javascript enabled.
  - Internet Explorer® 7 (or later)
  - Firefox™ 3.0 (or later)
  - Google Chrome 10 (or later)

  If browser security is set to **High**, you should add ECLWatch as a Trusted Site to allow Javascript execution.

- Install the ECL IDE

  The ECL IDE (Integrated Development Environment) is the tool used to create queries into your data and ECL files with which to build your queries.

  From the ECLWatch web page, download the Windows install set. If the link is not visible, either follow the link to the HPCC System's portal or install the Optional Packages.

  You can reach this page using the following URL:

  http://nnn.nnn.nnn.nnn:8010, where nnn.nnn.nnn.nnn is your node's IP address.

  The ECL IDE was designed to run on Windows machines. See the appendix for instructions on running on Linux workstations using Wine.

  - Microsoft VS 2008 C++ compiler (either Express or Professional edition). This is needed if you are running Windows and want to compile queries locally. This allows you to compile and run ECL code on your Windows workstation.
  - GCC. This is needed if you are running under Linux and want to compile queries locally on a standalone Linux machine, (although it may already be available to you since it usually comes with the operating system).
Follow these steps to install the packages and start components in a single-node configuration to begin. Once it is successfully installed, you will use the Configuration Manager to customize or expand your system.

Configuration Manager is the utility with which we configure the HPCC platform. It is run on your Linux Server and you access its interface using a browser.

**Figure 6. System Overview: Thor**
Figure 7. System Overview: Roxie
Initial Setup-Single Node

This section covers installing the HPCC on a single node. This will enable the HPCC system to operate successfully; however, the real strength of the HPCC is when it is run in a multi-node environment and can leverage the ability to perform operations using Massively Parallel Processing (MPP).

In addition, on a production system, you would dedicate one or more nodes to each server process. See the Using Config Manager manual for more details.

Installing the Package

The installation and package that you download is different depending on the operating system you plan to use. The installation packages will fail to install if their dependencies are missing from the target system. The dependencies for each operating system are listed in Specific OS node software installation commands in the Appendix.


To install the package, follow the appropriate installation instructions:

Centos/Red Hat/SuSe

Install RPM with the -Uvh switch.

This is the upgrade command and will perform an automatic upgrade if a previous version is installed or it will just install fresh if no other version has been installed.

```
sudo rpm -Uvh <rpm file name>
```

Ubuntu

For Ubuntu installations a Debian package is provided. To install the package, use:

```
sudo dpkg -i <deb filename>
```

Installing the Optional Packages

There are three optional packages you can install which will enable resources to be available from your ESP server's ECL Watch page. These resources include:

- Documentation
- ECL IDE
- Graph Control

We recommend installing these packages on your ESP Server node.

You can find these packages at http://hpccsystems.lexisnexis.com/download/free-community-edition
To install the package, follow the appropriate installation instructions:

**Centos/Red Hat/SuSe**

Install RPM with the -Uvh switch.

This is the upgrade command and will perform an automatic upgrade if a previous version is installed or it will just install fresh if no other version has been installed.

`sudo rpm -Uvh <rpm file name>`

**Ubuntu**

For Ubuntu installations a Debian package is provided. To install the package, use:

`sudo dpkg -i <deb filename>`

**Initial Startup**

1. Start the system using the default configuration.

**Centos/Red Hat/SuSe**

`sudo /sbin/service hpcc-init start`

**Ubuntu**

`sudo service hpcc-init start`

There are log files for each component in directories below `/var/log/HPCCSystems` (default location). If any component fails to start, these logs can help in troubleshooting.
Running an ECL Query on your Single-Node System

The single node system is running, and you can now create and run some ECL\(^1\) code using either ECL IDE the command line ECL compiler, or the ECLPlus tool.

Install the ECL IDE and HPCC Client Tools

1. In your browser, go to the ECL Watch URL For example, http://nnn.nnn.nnn.nnn:8010, where nnn.nnn.nnn.nnn is your node's IP address.

Your IP address could be different from the ones provided in the example images. Please use the IP address of your node.

2. From ECL Watch page, click on the Resources link in the menu on the left side.

Figure 8. ECL Watch Resource Page

---

\(^1\)Enterprise Control Language (ECL) is a declarative, data centric programming language used to manage all aspects of the massive data joins, sorts, and builds that truly differentiate HPCC (High Performance Computing Cluster) from other technologies in its ability to provide flexible data analysis on a massive scale.

© 2011 HPCC Systems. All rights reserved
Note: The order of items on the page may vary depending on the order in which optional packages are installed.

If this page is not visible, either follow the link to the HPCC System's portal or install the Optional Packages,

3. Click on the ECL IDE Installer link. When prompted save this file to your PC and then run it—do not run directly from your browser. This is a Windows installer for a Windows application.

4. You can close your browser, if desired.

5. Install the ECL IDE, following the prompts in the installer. Once the ECL IDE is installed successfully, you can proceed.
Running a basic ECL program

Now that the package is installed on your Linux node and ECL IDE is installed on your Windows workstation, you can run your first ECL program. ECL programs may be run locally or remotely. For larger ECL jobs, you will want to target a remote cluster of machines, which may not be running the same operating system as the machine you are working on.

In this section we will use the command line interface to the compiler to compile and run ECL code locally.

The ECL compiler (eclcc) installs on to the eclcc server node when a package is installed. This should be in your path, so you can run it from anywhere on the server. It is also installed on a Windows machine when you install the ECL IDE. To compile and run on Windows, you also need the Visual Studio 2008 C++ compiler (see User Workstation Requirements for details).

1. Create a file called hello.ecl and type in the following text (including the quotes):

```ecl
output('Hello world');
```

You can either use your favorite editor, or you can use the command line by typing the following

```bash
echo "Output('Hello world');" > hello.ecl
```

2. Compile your program using eclcc by typing the following command:

```bash
eclcc hello.ecl
```

3. An executable file is created which you can run as follows:

```bash
# on a Linux machine:
./a.out
# on a Windows machine:
a.out
```

This generates the output "Hello world" (excluding quotes), to the std output, your terminal window in this example. You can redirect or pipe the output to a file or program if you choose. This verifies that the compiler is working properly.

Running remotely using ECLPlus

The ECLPlus application accepts command line parameters to send directly to an ECL execution engine. You can use eclplus to control the creation and execution of larger ECL jobs which target a remote system. To compile jobs on a remote system, eclcc is used to create an archive of the ECL code to be compiled, and eclplus is used to submit it to a queue for compilation by the remote compiler server (eclccserver).

To submit a job to the queue using ECL Plus, make sure the HPCC has been started and use the following syntax:

```bash
eclplus @hello.ecl cluster=hthor server=<IP Address of the ESP node>:8010
```

or

```bash
eclplus @hello.ecl cluster=hthor server=
```

Where "." indicates the IP of the current box.

The workunit ID, status, and result are shown at the command line.

---

2. A Workunit is a record of a task submitted to an HPCC. It contains an identifier--workunit ID, the ECL code, results, and other information about the job.
View the full details of the workunit using the ECL Watch interface for your HPCC at this location http://nnn.nnn.nnn.nnn:8010, where nnn.nnn.nnn.nnn is the IP of your node. Either search for the workunit using the workunit ID or select ECL Workunits/Browse and find your workunit in the list provided.

Setting up an `eclplus.ini` file makes running a workunit a little easier when you want to use the same settings every time you submit a workunit in this way. See the *HPCC Client Tools* manual for details.

If your ECL is more complex than a single source file, you can use the eclcc compiler locally to create an archive to be sent to the eclccServer:

```
eclcc hello.ecl -E | eclplus @ cluster=thor server=<IP Address of the ESP>:8010
```

The cluster parameter must name a valid target cluster name as listed in your environment's topology section.

### Running a basic ECL program from the ECL IDE

1. Open the ECL IDE on your Windows workstation, from your start menu. (Start ► All Programs ► HPCCSystems ► ECL IDE).

   ![ECL IDE window](image)

   You can create a shortcut on your desktop to provide quick access to the ECL IDE.

2. Enter the **Login ID** and **Password** provided in the Login dialog.

<table>
<thead>
<tr>
<th>Login ID</th>
<th>hpcdemo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>hpcdemo</td>
</tr>
</tbody>
</table>

   **Figure 9. Login Window**

3. Open a new **Builder Window** (CTRL+N) and write the following code:

   ```ecl
   OUTPUT('Hello World');
   ```

   This could also be written as:

   ```ecl
   'Hello World';
   ```

   In the second program listing, the `OUTPUT` keyword is omitted. This is possible because the language is declarative and the `OUTPUT` action is implicit.

4. Select **thor** as your target cluster.

   **Thor** is the Data Refinery component of your HPCC. It is a disk based massively parallel computer cluster, optimized for sorting, manipulating, and transforming massive data.
5. Press the syntax check button on the main toolbar (or press F7).
Figure 11. Syntax Check

A successful syntax check displays the "No Errors" message.

6. Press the Go button (or press ctrl+enter).
Figure 12. Completed job

The green check mark indicates successful completion.

7. Click on the workunit number tab and then on the Result 1 tab to see the output.
Figure 13. Completed job output

```
## Result 1
1 Hello World
```

No Errors...
Configuring a Multi-Node System

While the single-node system is fully-functional, it does not take advantage of the true power of an HPCC—the ability to perform operations using Massively Parallel Processing (MPP). This section provides the steps to expand your single-node system into a multi-node system using the Configuration Manager Wizard.

Using the Configuration Manager Wizard

This section details reconfiguring a system to use multiple nodes. Before you start this section, you must have already downloaded the correct packages for your distro from the HPCC Systems website: http://hpccsystems.com/download/free-community-edition

1. If it is running, stop the HPCC system, using this command:

   **Centos/Red Hat/SuSe**
   
   ```
   sudo /sbin/service hpcc-init stop
   ```

   **Ubuntu**
   
   ```
   sudo service hpcc-init stop
   ```

   You can use this command to confirm HPCC processes are stopped (on Centos/Red Hat/SuSe):
   
   ```
   sudo /sbin/service hpcc-init status
   ```

   For Ubuntu
   
   ```
   sudo service hpcc-init status
   ```

2. Start the Configuration Manager service.

   ```
   sudo /opt/HPCCSystems/sbin/configmgr
   ```

3. Leave this window open. You can minimize it, if desired.

4. Using a Web browser, go to the Configuration Manager's interface:

   ```
   http://<node_ip>:8015
   ```
5. The Configuration Manager startup wizard displays. To use the wizard, select the Generate new environment using wizard button.

![Configuration Manager Startup Wizard](image)

6. Provide a name for the environment file.

   This will then be the name of the configuration xml. For example, we will name this `NewEnvironment.xml`.

7. Press the `Next` button.

   Next you will need to define the IP addresses that your system will use.

8. Enter the all the IP addresses you want to use in this HPCC.

   The IP addresses do not need to be contiguous. In the image below, we specified the IP addresses nn.nnn.nnn.1-100 and nn.nnn.nnn.111. These are separated with a semi-colon.

   You can specify a range of IPs using a hyphen (for example, NNN.NNN.NNN.1-100). IP Addresses can be specified individually using semi-colon delimiters.
9. Press the **Next** button.

   Alternatively, you could find the IP addresses using Auto Discovery by selecting the Auto Discovery button. (only available in Enterprise Edition)

   Now you will define how many nodes to use for the Roxie and Thor clusters.

10. Enter the appropriate values as indicated.
11. Press the Next button

The Environment Summary displays.

12. Click on Finish to accept these values. This saves the file.

Keep in mind, that your HPCC configuration may be different depending on your needs. For example, you may not need a Roxie or you may need several smaller Roxie clusters. In addition, in a production [Thor] system, you would ensure that Thor and Roxie nodes are dedicated and have no other processes running on them. This document is intended to show you how to use the configuration tools. Capacity planning and system design is covered in a training module.
13. You will now be notified that you have completed the wizard.

Successfully generated the file

NewEnvironment.xml
At this point the system has created a file named NewEnvironment.xml in the /etc/HPCCSystems/source directory

14. Stop the Configuration Manager in the terminal where you started it by pressing CTRL-C.

Be sure system is stopped before attempting to move the environment.xml file.

15. Copy the NewEnvironment.xml file from the source directory to the /etc/HPCCSystems and rename the file to environment.xml

# for example
sudo -u hpcc cp /etc/HPCCSystems/source/NewEnvironment.xml /etc/HPCCSystems/environment.xml

Make sure that you have sufficient privileges to write file(s) to the destination directory before attempting to copy. If prompted to overwrite the destination file, you should answer yes. The environment.xml file MUST be owned by the hpcc user.

16. If you have added new machines to the cluster, you need to copy and install the HPCC package onto all nodes, and generate and clone the SSH keys. This can be done using the install-cluster.sh script which is provided with HPCC. Use the following command:

/opt/HPCCSystems/sbin/install-cluster.sh -k <package-file-name>

Where <package-file-name> is the name of the package file that you want to install on every node - this will be in the form hpccsystems-platform-xxxx-n.n.nnnn.rpm (or .deb) depending on the version and distro. More details including other options that may be used with this command are included in the appendix.

17. Copy the /etc/HPCCSystems/environment.xml to /etc/HPCCSystems/ on every node.

You may want to create a script to push out XML file to all nodes. A sample script is provided with HPCC. The following command copies the XML files out to all nodes as required:

sudo -u hpcc /opt/HPCCSystems/sbin/hpcc-push.sh <sourcefile> <destinationfile>

See the appendix for more information on using this script.

18. Restart the HPCC system on every node. The following command starts the HPCC system on an individual node:

**Centos/Red Hat/SuSe**

sudo /sbin/service hpcc-init start

**Ubuntu**

sudo service hpcc-init start

You may want to create a script to push this command out to every node. A sample script is provided with HPCC. Use the following command to start HPCC on all nodes:

sudo -u hpcc /opt/HPCCSystems/sbin/hpcc-run.sh -a hpcc-init start

This script can also be used to stop HPCC on all nodes and to stop and start individual components on all nodes. See the appendix for more details.
Starting and Stopping

Start, Stop, Restart the System

Once you have your system environment established, the init system can be used to start, stop, or restart components.

The following commands can be used:

To start the system:

**Centos/Red Hat/SuSe**

```bash
sudo /sbin/service hpcc-init start
```

**Ubuntu**

```bash
sudo service hpcc-init start
```

To stop the system:

**Centos/Red Hat/SuSe**

```bash
sudo /sbin/service hpcc-init stop
```

**Ubuntu**

```bash
sudo service hpcc-init stop
```

You can use a script to start or stop multiple nodes in the system. See Example Scripts in the Appendix section for samples.

To start or stop a single component, you can use the -c flag in the init system as follows.

**Centos/Red Hat/SuSe**

```bash
sudo /sbin/service hpcc-init -c <component name> <command>
```

**Ubuntu**

```bash
sudo service hpcc-init -c <component name> <command>
```

To stop dafilesrv (a helper application), you must use this command: sudo /sbin/service dafilesrv stop. See Helper Applications for details.

Start or Stop Configuration Manager

Configure the system as desired using Configuration Manager.

1. If the system is running, stop the HPCC system, using this command on every node:

**Centos/Red Hat/SuSe**

```bash
sudo /sbin/service hpcc-init stop
```

**Ubuntu**

```bash
sudo service hpcc-init stop
```
2. Start the Configuration Manager service on one node (usually the first node is considered the head node and is used for this task, but this is up to you)

```
sudo /opt/HPCCSystems/sbin/configmgr
```

3. Using a web browser, go to the Configuration Manager's interface:

```
http://<ip of installed system>:8015
```
More Examples

This section contains additional ECL examples you can use on your HPCC cluster. You can run these on a single-node system or a larger multi-node cluster.

ECL Example: Anagram1

This example takes a STRING and produces every possible anagram from it. This code is the basis for a second example which evaluates which of these are actual words using a word list data file.

1. Open the ECL IDE (Start ► All Programs ► HPCC Systems ► ECL IDE ) and login to your HPCC.

2. Open a new Builder Window (CTRL+N) and write the following code:

   ```
   STRING Word := 'FRED' : STORED('Word');
   R := RECORD
       STRING SoFar {MAXLENGTH(200)};
       STRING Rest {MAXLENGTH(200)};
   END;
   Init := DATASET([{'',Word}],R);
   R Pluck1(DATASET(R) infile) := FUNCTION
       R TakeOne(R le, UNSIGNED1 c) := TRANSFORM
           SELF.SoFar := le.SoFar + le.Rest[c];
           SELF.Rest := le.Rest[..c-1]+le.Rest[c+1..];
       // Boundary Conditions handled automatically
       END;
   RETURN NORMALIZE(infile,LENGTH(LEFT.Rest),TakeOne(LEFT,COUNTER));
   END;
   L := LOOP(Init,LENGTH(TRIM(Word)),Pluck1(ROWS(LEFT)));
   OUTPUT(L);
   ```

3. Select thor as your target cluster.

4. Press the syntax check button on the main toolbar (or press F7)

5. Press the Submit button (or press ctrl+enter).
Figure 14. Completed job

The green check mark indicates successful completion.

6. Click on the workunit number tab and then on the Result 1 tab to see the output.
Figure 15. Completed job output

<table>
<thead>
<tr>
<th>#</th>
<th>solar</th>
<th>rest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FEED</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FEDE</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>FFRD</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>FEDR</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>FDRE</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>FDER</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RFED</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>RFDE</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>RFDD</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>REDF</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>RDFE</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>RDEF</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>EFRD</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>EFDR</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>EFDD</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>EERD</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>ELFR</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>EDRF</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>DFRE</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>DFER</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>DEFE</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>DREF</td>
<td></td>
</tr>
</tbody>
</table>
Roxie Example: Anagram2

In this example, we will download an open source data file of dictionary words, spray that file to our Thor cluster, then validate our anagrams against that file so that we determine which are valid words. The validation step uses a JOIN of the anagram list to the dictionary file. Using an index and a keyed join would be more efficient, but this serves as a simple example.

Download the word list

We will download the word list from http://wordlist.sourceforge.net/

1. Download the Official 12 Dicts Package. The files are available in tar.gz or ZIP format.

2. Extract the 2of12.txt file to a folder on your local machine.

Load the Dictionary File to your Landing Zone

In this step, you will copy the data files to a location from which it can be sprayed to your HPCC cluster. A Landing Zone is a storage location attached to your HPCC. It has a utility running to facilitate file spraying to a cluster.

For smaller data files, maximum of 2GB, you can use the upload/download file utility in ECL Watch. This data file is only ~400 kb.

Next you will distribute (or Spray) the dataset to all the nodes in the HPCC cluster. The power of the HPCC comes from its ability to assign multiple processors to work on different portions of the data file in parallel. Even though the VM Edition only has a single node, the data must be sprayed to the cluster.

1. In your browser, go to the ECL Watch URL For example, http://nnn.nnn.nnn.nnn:8010, where nnn.nnn.nnn.nnn is your ESP Server's IP address.

   Your IP address could be different from the ones provided in the example images. Please use the IP address provided by your installation.

2. From ECL Watch page, click on the Upload/download File link in the menu on the left side.
Once you click on the Upload/download file link, it will take you to the Dropzones and Files page, where you can choose to Browse your machine for a file to upload:

3. Press the Browse button to browse the files on your local machine, select the file to upload and then press the Open button.
The file you selected should appear in the **Select a file to upload** field. The data file is named: **2of12.txt**.

4. Press on **Upload Now** to complete the file upload.

## Spray the Data File to your **Data Refinery (Thor) Cluster**

To use the data file in our HPCC system, we must “spray” it to all the nodes. A *spray* or *import* is the relocation of a data file from one location (such as a Landing Zone) to multiple file parts on nodes in a cluster.

The distributed or sprayed file is given a *logical-file-name* as follows: `~thor::word_list_csv` The system maintains a list of logical files and the corresponding physical file locations of the file parts.

1. Open ECL Watch using the following URL:

   http://nnn.nnn.nnn.nnn:pppp(where nnn.nnn.nnn.nnn is your ESP Server’s IP Address and pppp is the port. The default port is 8010)

2. Click on the Spray CSV hyperlink under the DFU Files menu on the left.

   ![Figure 18. Spray CSV](image)

   The DFU Spray CSV page displays.

3. Select mydropzone in the Source **Machine/dropzone** drop-list.
The IP Address is automatically filled and the Local Path is partially filled with the default folder on your landing zone. Note: The VM and Community Edition typically only has one landing zone defined.

4. Complete the Local Path to include the complete file name or use the Choose File button to select the file from a list of files in the folder. The file is 2of12.txt.

5. Fill in the rest of the parameters (if they are not filled in already).
   - Max Record Length 8192
   - Separator \
   - Line Terminator \n,\r\n
6. Fill in the Label using the Logical File name desired: thor::word_list_csv

7. Make sure the Overwrite and Replicate boxes are checked.

Figure 19. Spray the File

8. Press the Submit button
Figure 20. View Progress

9. Click the View Progress link

10. The Workunit progress page displays.

Figure 21. Workunit Progress
Run the query on Thor

1. Open a new **Builder Window** (CTRL+N) and write the following code:

```hpcc
IMPORT Std;
layout_word_list := record
  string word;
end;
File_Word_List := dataset('~thor::word_list_csv', layout_word_list,
  CSV(heading(1),separator(','),quote('')));
STRING Word := 'teacher' :STORED('Word');
STRING SortString(STRING input) := FUNCTION
  OneChar := RECORD
    STRING c;
  END;
  OneChar MakeSingle(OneChar L, unsigned pos) := TRANSFORM
    SELF.c := L.c[pos];
  END;
  Split := NORMALIZE(DATASET([input],OneChar), LENGTH(input),
    MakeSingle(LEFT,COUNTER));
  SortedSplit := SORT(Split, c);
  OneChar Recombine(OneChar L, OneChar R) := TRANSFORM
    SELF.c := L.c+R.c;
  END;
  Recombined := ROLLUP(SortedSplit, Recombine(LEFT, RIGHT),ALL);
  RETURN Recombined[1].c;
END;
STRING CleanedWord := SortString(TRIM(Std.Str.ToUpperCase(Word)));
R := RECORD
  STRING SoFar {MAXLENGTH(200)};
  STRING Rest {MAXLENGTH(200)};
END;
Init := DATASET([{'',CleanedWord}],R);
R Pluck1(DATASET(R) infile) := FUNCTION
  R TakeOne(R le, UNSIGNED1 c) := TRANSFORM
    SELF.SoFar := le.SoFar + le.Rest[c];
    SELF.Rest := le.Rest[..c-1]+le.Rest[c+1..];
    // Boundary Conditions
    // handled automatically
  END;
  RETURN DEDUP(NORMALIZE(infile,LENGTH(LEFT.Rest),TakeOne(LEFT,COUNTER)));
END;
L := LOOP(Init,LENGTH(CleanedWord),Pluck1(ROWS(LEFT)));
ValidWords := JOIN(L,File_Word_List,
  LEFT.SoFar=Std.Str.ToUpperCase(RIGHT.Word),TRANSFORM(LEFT));
OUTPUT(CleanedWord);
COUNT(ValidWords);
OUTPUT(ValidWords)
```

2. Select **thor** as your target cluster.

3. Press the syntax check button on the main toolbar (or press F7)

4. Press the **Submit** button.

5. When it completes, select the Workunit tab, then select the Result tab.

6. Examine the result.
Compile and Publish the query to Roxie

1. RT-CLICK on the My Files folder in the Repository window, and select Insert Folder from the pop-up menu.

Figure 22. Insert Folder

2. Enter Anagram for the label, then press the OK button.
3. RT-CLICK on the Anagram Folder, and select Insert File from the pop-up menu.

4. Enter ValidateAnagrams for the label, then press the OK button.

A Builder Window opens.

Figure 24. Builder Window

5. Write the following code (you can copy the code from the other builder window):

```ecl
IMPORT Std;
layout_word_list := record
  string word;
end;
File_Word_List := dataset('~thor::word_list_csv', layout_word_list,
  CSV(heading(1),separator(','),quote('')));
STRING Word := 'teacher' :STORED('Word');
STRING SortString(STRING input) := FUNCTION
  OneChar := RECORD
    STRING c;
  END;
  OneChar MakeSingle(OneChar L, unsigned pos) := TRANSFORM
    SELF.c := L.c[pos];
```
END;
Split := NORMALIZE(DATASET([input],OneChar), LENGTH(input),
MakeSingle(LEFT,COUNTER));
SortedSplit := SORT(Split, C);
OneChar Recombine(OneChar L, OneChar R) := TRANSFORM
SELF.c := L.c+R.c;
END;
Recombined := ROLLUP(SortedSplit, Recombine(LEFT, RIGHT), ALL);
RETURN Recombined[1].c;
END;

STRING CleanedWord := SortString(TRIM(Std.Str.ToUpperCase(Word)));

R := RECORD
  STRING SoFar {MAXLENGTH(200)};
  STRING Rest {MAXLENGTH(200)};
END;
Init := DATASET(["",CleanedWord], R);
R Pluck1 (DATASET(R) infile) := FUNCTION
  R TakeOne(R le, UNSIGNED1 c) := TRANSFORM
    SELF.SoFar := le.SoFar + le.Rest[c];
    SELF.Rest := le.Rest[..c-1]+le.Rest[c+1..];
    // Boundary Conditions
    // handled automatically
END;
RETURN DEDUP(NORMALIZE(infile, LENGTH(LEFT.Rest), TakeOne(LEFT, COUNTER)));
END;
L := LOOP(Init, LENGTH(CleanedWord), Pluck1(ROWS(LEFT)));
ValidWords := JOIN(L, File_Word_List,
  LEFT.SoFar=Std.Str.ToUpperCase(RIGHT.Word), TRANSFORM(LEFT));
OUTPUT(CleanedWord);
COUNT(ValidWords);
OUTPUT(ValidWords)

6. Select **Roxie** as your target cluster.

7. Press the syntax check button on the main toolbar (or press F7)

8. In the Builder window, in the upper left corner the **Submit** button has a drop down arrow next to it. Select the arrow to expose the **Compile** option.

Figure 25. Compile

9. Select **Compile**

10. When it completes, select the Workunit tab, then select the Result tab.
11. When the workunit finishes, it will display a green circle indicating it has compiled.

**Figure 26. Compiled**

![Compiled](image)

**Publish the Roxie query**

Next we will publish the query to a Roxie Cluster.

1. Select the workunit tab for the ValidateAnagrams that you just compiled.

2. Select the ECL Watch tab.

3. Press the **Publish** button (you may need to scroll down the main window)
Figure 27. Publish Query

When it successfully publishes, you will see:
Run the Roxie Query in WsECL

Now that the query is published to a Roxie cluster, we can run it using the WsECL service. WsECL is a web-based interface to queries on an HPCC platform. Use the following URL:

http://nnn.nnn.nnn.nnn:pppp (where nnn.nnn.nnn.nnn is your ESP Server’s IP address and pppp is the port. The default port is 8002)

1. Click on the + sign next to myroxie to expand the tree.

2. Click on the ValidateAnagrams.1 hyperlink.

   The form for the service displays.

3. Select Output Tables in the drop-list.

4. Provide a word to make anagrams from (e.g., TEACHER), then press the Submit button.

   The results display.
Figure 30. RoxieResults

validateanagrams.1 Response

Dataset: Result 1

Result 1
1 ACEEHR

Dataset: Result 2

Result 2
1 4

Dataset: Result 3

sofar  rest
1 CHEATER
2 HECTARE
3 RETEACH
4 TEACHER
Next Steps

Available from the menu in the ECL IDE there are several documents which provide details on various aspects of the HPCC.

You can access them from the help menu: Help ► Documentation.

**Figure 31. Help Menu**

You can also find these from the **Start** menu:

Start ► All Programs ► HPCC Systems ► ECL IDE ► Docs

To familiarize yourself with what your system can do we recommend following the steps in

- The **HPCC Data Tutorial**
- **The Six Degrees of Kevin Bacon** example
- Read **Using Config Manager** to learn how to configure an HPCC platform using Advanced View.
- Use your new skills to process your own massive dataset!

The HPCC Systems Portal is also a valuable resource for more information including:

- Video Tutorials
- Additional examples
- White Papers
- Documentation
Example Scripts

For a multi-node configuration, the packages must be installed on each node. You can install each one manually or use scripts to copy and install the packages. On a large system where you have many nodes copying and installing on every node is not practical, therefore we provide some scripts you can use or to serve as examples to give you a start in making your own.

Scripts are installed to the `/opt/HPCCSystems/sbin` directory.

Make sure that you have the sufficient privileges to sudo as an administrator to use the `install-cluster.sh` script. To use the `hpcc-push.sh` or `hpcc-run.sh` scripts, you must sudo as user `hpcc`.

**install-cluster.sh**

`install-cluster.sh [-k] <package-name>`

- `<package-name>` Name of the HPCC package to install. Required
- `-k` When specified, the script generates and distributes ssh keys to all hosts. Optional.

You can run this script as any user with sufficient permissions to execute it; however, when prompted for username/password, you must provide credentials for a user with sufficient sudo rights to run commands as an administrator on all nodes.

Before you can use this script, you must have already defined and generated an environment.xml file (using ConfigMgr's wizard or advanced mode). This script:

- reads the active environment.xml file and gathers a list of nodes upon which to act.
- installs the HPCC platform package(s) on all nodes specified.
- pushes out and deploys the environment file (environment.xml) to all nodes specified.
- optionally, if you specify the `-k` option it also generates the required ssh keys and deploys them as required to all nodes specified.
Examples:

This example installs the HPCC Platform packages to remaining nodes and pushes out the active environment.xml file to those nodes:

```
/opt/HPCCSystems/sbin/install-cluster.sh hpccsystems-platform-xxxx-n.n.nnnn
```

(where \( n.n.nnnn \) is the build number)

This example installs the HPCC Platform packages to all nodes and pushes out the active environment.xml file to those nodes. It also generates ssh keys and pushes them out to all nodes.

```
/opt/HPCCSystems/sbin/install-cluster.sh -k hpccsystems-platform-xxxx-n.n.nnnn
```

(where \( n.n.nnnn \) is the build number)
**hpcc-push.sh**

To use this script, the ssh keys need to be properly configured on all nodes, and you must use sudo:

This script "pushes" files from the source filename and path to the destination filename and path for all IP addresses in the active environment.xml.

The IP addresses were defined when editing the environment in ConfigMgr.

```bash
sudo -u hpcc /opt/HPCCSystems/sbin/hpcc-push.sh <sourcefile> <destinationfile>
```

For example:

```bash
sudo -u hpcc /opt/HPCCSystems/sbin/hpcc-push.sh \
/etc/HPCCSystems/environment.xml /etc/HPCCSystems/environment.xml
```
hpcc-run.sh

\texttt{hpcc-run.sh \{-c component\} \{-a \{hpcc-init|dafilesrv\}\} \{start|stop|restart|status|setup\}}

\texttt{-c} \hspace{1em} \bullet \hspace{1em} \texttt{-c componentname}

Specifies the component upon which to execute the command. If omitted, the default is \textbf{all} components on the machine.

\texttt{-c componenttype}

Specifies the component type upon which to execute the command. If more than one of this type is configured, all will be acted upon. If omitted, the default is \textbf{all} components on the machine.

\texttt{-a} \hspace{1em} \begin{itemize}
    \item \texttt{hpcc-init: [start|stop|restart|status|setup]}
    \item \texttt{dafilesrv [start|stop]}
\end{itemize}

To use this script, the ssh keys need to be properly configured on all nodes, and you must \texttt{sudo} as user hpcc:

\textbf{This script runs a command on all IP addresses in the active environment.xml.}

The IP addresses were defined when editing the environment in ConfigMgr. This script supports all the parameters of \texttt{hpcc-init} and \texttt{dafilesrv}.

\textbf{Example:}

This example starts all components on the nodes

\texttt{sudo -u hpcc /opt/HPCCSystems/sbin/hpcc-run.sh \{-a hpcc-init\} start}

This example starts all components of the esp type on the nodes

\texttt{sudo -u hpcc /opt/HPCCSystems/sbin/hpcc-run.sh \{-c esp\} \{-a hpcc-init\} start}

This example starts all components with a component name myesp on the nodes

\texttt{sudo -u hpcc /opt/HPCCSystems/sbin/hpcc-run.sh \{-c myesp\} \{-a hpcc-init\} start}

This example starts the dafilesrv helper application

\texttt{sudo -u hpcc /opt/HPCCSystems/sbin/hpcc-run.sh \{-a dafilesrv\} start}
Uninstalling the HPCC Platform

To uninstall the HPCC platform from your system, issue the following command. If necessary, do so on each node that it is installed on.

```
sudo rpm -e HPCCSystems-platform-xxxx-n.n.nnnn
```

(where $n.n.nnnn$ is the build number)
Specific OS node software Installation commands:

The following are some Operating System specific commands or options which may differ from what is referred to in this manual.

**CentOS/Red Hat**

The following packages are required dependencies and may need installing:

- yum install libicu
- yum install gcc-c++
- yum install xalan-c
- yum install boost141-regex-1.41.0-2.el5.x86_64.rpm
- yum install openssh-server
- yum install openssh-client
- yum install openldap
- yum install m4
- yum install libtool --- may have already been installed as part of another package
- yum install xerces-c --- may have already been installed as part of another package
- yum install openssl --- may have already been installed as part of another package
- yum install expect

**Ubuntu commands**

The following packages are required dependencies and may need installing:

- apt-get install libicu-dev
- apt-get install libboost-regex-dev
- apt-get install libxalan110
- apt-get install libxerces-c28
- apt-get install openssl
- apt-get install binutils
- apt-get install g++
- apt-get install libldap2-dev
- apt-get install zlib1g
- apt-get install openssh-server
- apt-get install openssh-client
- apt-get install expect

If this is the first time you are installing the HPCC System on Ubuntu, all dependencies may need to be installed. To do this in one step use the following command:

```
sudo apt-get -f install
```
SuSe

The following packages are required dependencies and may need installing:

- zypper in libicu
- zypper in libboost-regex1_42_0 (SuSe 11.3)
- zypper in libboost-regex1_44_0 (SuSe 11.4)
- zypper in libxalan-c110
- zypper in libxerces-c-3_0
- zypper in binutils
- zypper in gcc-c++
- zypper in libldap-2_4-2
- zypper in openssh
- zypper in expect

Additional steps may need to be taken if you have iptables firewall restrictions enabled on your SuSe system. The iptables firewall restrictions could cause issues for the HPCC system. You should either disable iptables or add rules for access based on the ports configured for each component in configmgr.

For example: to allow access to ECL Watch use the following command:

`iptables -A INPUT -p tcp --dport 8010 -j ACCEPT`

To remove this rule when not wanted, use the following command:

`iptables -D INPUT -p tcp --dport 8010 -j ACCEPT`

Similarly you would have to add additional rules for other components or ports.
Helper Applications

There is a helper application that runs on all nodes that you may need to stop or start manually.

Normally, this process is started automatically the first time the hpcc-init service executes.

Enter the following commands to stop or start the helper application:

- dafilesrv

  ```bash
  sudo /sbin/service dafilesrv stop
  sudo /sbin/service dafilesrv start
  ```
hpcc-init

/sbin/service hpcc-init [option] command

option
  • -c componentname, --component=componentname

Specify the component upon which to execute the command. If omitted, the default is all components on the machine.

- c componenttype, --component=componenttype

Specify the component type upon which to execute the command. If more than one of this type is configured, all will be acted upon. If omitted, the default is all components on the machine.

• --componentlist

Provides a list of all component names on the current node as specified in the environment file.

• --typelist

Provides a list of all component types on the current node as specified in the environment file.

• -h, --help

Displays a help page

command
  • start:

Starts component(s)

• stop

Stops component(s)

• status

Displays component(s) status

• restart

Restarts component(s)

• force-reload

Deletes all local configuration files, data files, log files, and then restarts component(s). BE CAREFUL using this command.

• setup

Initializes component configuration files but does not start the component(s).

The hpcc-init function is used to start, stop, restart, setup, or check the status of any or all HPCC components.
Examples:

```bash
sudo /sbin/service hpcc-init start
sudo /sbin/service hpcc-init stop

sudo /sbin/service hpcc-init -c myeclserver start
sudo /sbin/service hpcc-init --component=myeclserver start

sudo /sbin/service hpcc-init -c esp start
```
Running the ECL IDE under WINE

To run the ECL IDE under WINE in Linux, follow these steps.

1. Install wine1.2 (this corresponds to Wine version 1.1.31) and its dependencies.

2. Download msxml3.msi from Microsoft (Service Pack 7 or later).
   
   http://support.microsoft.com/kb/308480/en-us

3. Install msxml3.msi in Wine (DOUBLE-CLICK the msi file and Wine will install it).

4. Open Configure Wine (Applications/Wine/Configure Wine):

5. Select the Libraries tab.

6. In the New override for library droplist, select msxml3, then press the add button.

7. Select msxml3 in the Existing overrides list and press Edit.

8. Select the Native (Windows) option and press the OK button.

9. Press the OK button to close the Wine Configuration window.

10. Install the HPCC ECL IDE (DOUBLE-CLICK the setup.msi file and Wine will install it).